FieldServer Technologies
A Sierra Monitor Corporation Company
1991 Tarob Court, Milpitas, California 95035 USA
Phone: (408) 262-2299
Toll Free: (888) 509-1970
Fax: (408) 262-2296
Email: support@fieldserver.com
Web Site: www.fieldserver.com

SlotServer Instruction Manual

APPLICABILITY & EFFECTIVITY

This manual provides instructions for the following FieldServer products:

Description
FS-RA-CLX-BAS
FS-RA-CLX-FIR
FS-RA-CLX-PRO

The instructions are effective for the above as of January 2006

Instruction Manual Part Number: T17011
Rev. A3.

TABLE OF CONTENTS

1. Introduction

1.1. About this product

The SlotServer Instruction Manual provides the information necessary to configure the SlotServer, allowing an Allen Bradley ControlLogix platform to pass data between a ControlLogix CPU and other third party communications protocols supported by the SlotServer. **The SlotServer uses implicit communications between the CPU and the SlotServer and is consequently treated as an I/O Server in RSLogix.**

The SlotServer Instruction Manual covers information for installing the SlotServer, and configuring the module to transfer data with the CPU on the ControlLogix Rack. Depending on the SlotServer Module ordered, supplementary driver manuals are provided for information on how to configure the third party protocols residing in the SlotServer.

2. Pre-Startup checklist

2.1. Accessories supplied with the SlotServer

Please verify that the following components were supplied with the SlotServer module:

FieldServer Technologies PART #	Description
	2-way combi-comb connector for LonWorks port
	Software CD
	CAT-5 Direct Ethernet Cable
	Documentation Binder

2.2. Third party components (not supplied by FieldServer Technologies)

The following components will need to be procured as a minimum for achieving ControlLogix integration. Ensure that these are available before commencing with commissioning

Part #	Description
AB1756-Ax	ControlLogix Rack
AB1756-L55/L61	ControlLogix CPU
AB1756-PA7x	ControlLogix PSU
9324-RLD300ENE	RSLOGIX 5000

2.3. Required configuration for the SlotServer

To achieve data transfer between CPU tags and the SlotServer third party protocols, it will be necessary to write and load a configuration into the SlotServer that tells the SlotServer how to map the ControlLogix Tags to the required protocol addresses. This configuration is written in a Comma Separated Variable (csv) file, and any text editor or spreadsheet program that supports csv format can be used for this purpose. FieldServer Technologies provides an example configuration file so that the configuration does not need to be written from scratch. Configuration parameters needed to exchange data between the CPU and the SlotServer data images (Data Arrays) are presented in Section 6. The appropriate driver manual supplement will describe how to map the data in and out of the Data Arrays for the relevant protocol.

The FieldServer configuration manual details basic and advanced techniques for the configuration of the SlotServer Manual, and it is strongly advised that this manual is read before attempting to write the SlotServer Configuration.

Note: FieldServer Technologies provides SlotServer configuration services if the user does not wish to perform the configuration themselves.

3. Steps for implementation of a SlotServer Project

3.1. Read the SlotServer Instruction Manual

This provides guidelines for getting started with the SlotServer, and is very useful for first time implementation of a SlotServer application.

3.2. Install the SlotServer in a ControlLogix Rack.

Refer to Section 4 for information on installation.

3.3. Write the SlotServer configuration.

- Refer to the FieldServer configuration manual for instruction on writing a FieldServer configuration.
- Refer to Section 6 for assistance on managing the Logix driver Parameters
- Refer to the desired protocol driver supplement for assistance on managing the protocol parameters.

3.4. Install the FieldServer Utilities

Insert the supplied Utilities CD into the CD drive of the PC to be used for configuration and follow the install instructions.

3.5. Load the SlotServer Configuration into the SlotServer

Refer to the FieldServer Utilities manual for information on the FieldServer Remote User Interface.

3.6. Program the ControlLogix CPU to communicate with the SlotServer.

The sections that follow highlight a few examples on how this is done.

3.7. Commission the third party network.

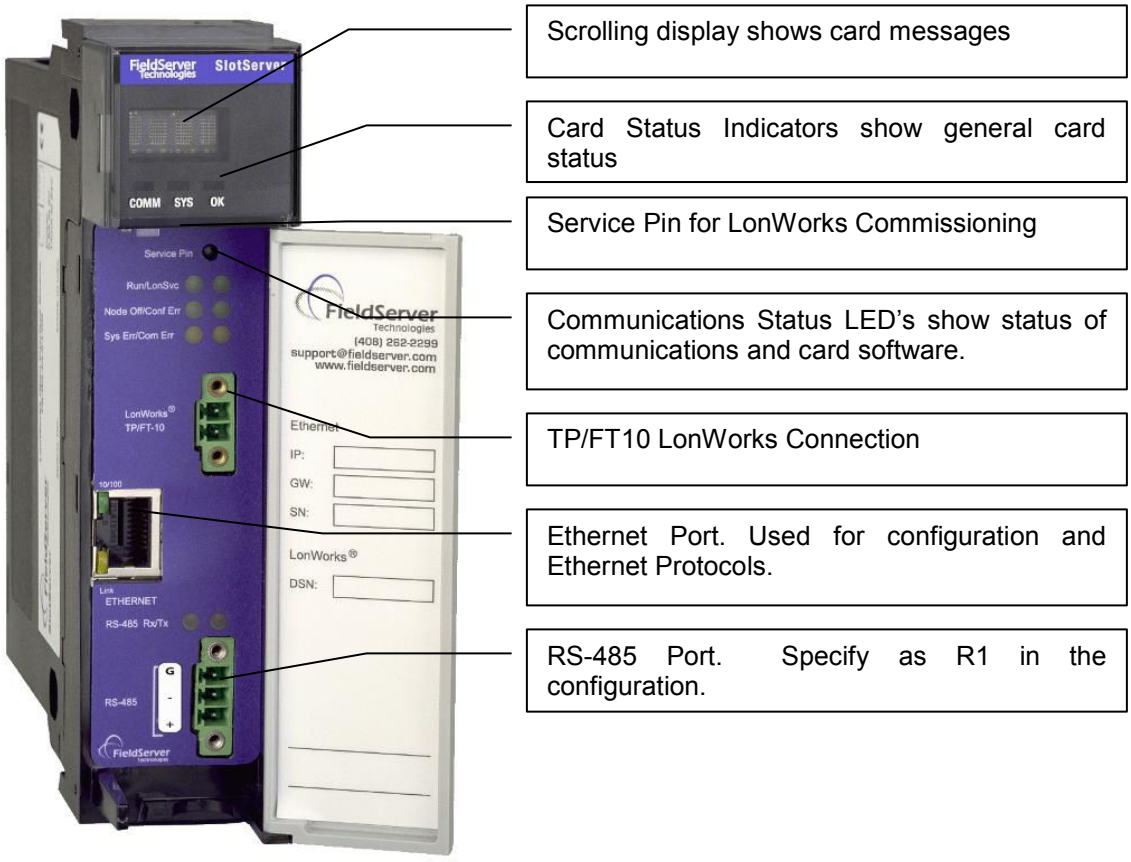
Refer to standard commissioning guidelines for the protocol in question.

4. SlotServer Installation

The SlotServer plugs directly into the AB1756 ControlLogix rack. Take note of the slot number used for the SlotServer as this will be needed when configuring the I/O in the RSLogix software.

4.1. SlotServer Ports and Displays

Note that there are several hardware ports available on the SlotServer card, but that this SlotServer model only makes use of a select few. The ports and displays of interest are as shown in the diagram below:

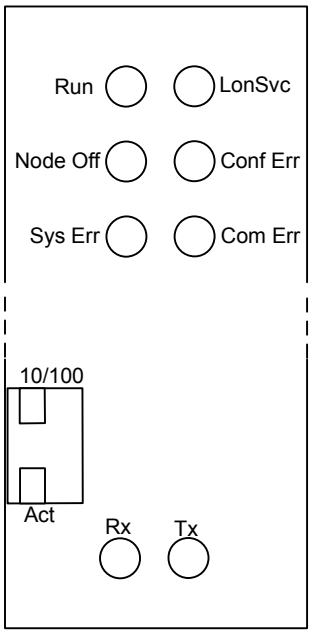


Scrolling Display
This will Display an OK status when the SlotServer has powered up successfully.

Card Status Indicators:
The Comm and OK lights will indicate green when the SlotServer has powered up successfully. The Sys indicator will indicate green if the external nodes are all communicating properly; and red otherwise.

Service Pin
To be used for commissioning the SlotServer into a LonWorks network

Communications Status LED's



LED	Description
Run	When the SlotServer is powered up, this light will start flashing after approximately 2 minutes. If this light does not flash, it is an indication that the firmware is not running, and that the SlotServer will not be operating.
LonSvc	This light will flash when the card's Neuron chip is unconfigured, and will extinguish when the SlotServer has been commissioned into a Network.
NodeOff	This light indicates that a device bound to the SlotServer on the LonWorks network is not communicating. Use the LonWorks Network Manager to troubleshoot the problem.
Conf Err	This light indicates an error in the configuration of the card. Since this model is shipped pre-configured, this light should never come on unless the card configuration is tampered with
Sys Err	This light indicates that an unexpected operation state occurred. The light can be reset by cycling power to the card. If the error re-occurs persistently, then contact FieldServer Technical support for assistance with troubleshooting after setting up the trade port as shown in Appendix A.
Com Err	This light indicates that a communication error occurred. Use the LonWorks Network Manager to troubleshoot the problem, or refer to Appendix B.1.
RS-485 (R1)	These lights are related to the RS-485 serial ports provided on the FieldServer. The **Rx** (Receive) light will flash red if the FieldServer is receiving data. The **Tx** (Transmit) light will flash red if the FieldServer is sending data. (Note that due to the nature of 2-wire RS-485 the Rx light will flash every time the Tx light flashes too). The frequency of the flashing is directly related to the frequency of data transfer.
Ethernet Lights (N1,N2)	These lights are related to the two Ethernet network ports provided on the FieldServer. The **10/100** light will be green if the physical connection to the network hub is running at 100 MBits/s. If the light is off, the connection is running at 10 MBits/s. The **Act** (Activity) light will indicate activity on the network. The frequency of flashing is directly related to the network activity. It does not necessarily mean data is being received by the FieldServer. Note that Ethernet Port N2 supports 10 or 100 MBits/s, whereas N1 only supports 10 MBits/s

TP/FT10 LonWorks connection
This two wire connection is polarity insensitive and can be multi-dropped into a LonWorks Network.

Ethernet Port
The Ethernet port can be used for configuration and troubleshooting, as well as for Protocols that require an Ethernet connection. Use standard Ethernet cables for this connection.

RS-485 Port
This is a standard 2-wire RS-485 port. Specify as R1 in the configuration.

4.2. Inside Cover Door Label

A label has been provided on the inside of the cover door for the User to document important or useful configuration parameters used. In particular, special provision has been made to document the Ethernet port IP parameters, as well as the LonWorks DSN (Domain/Subnet/Node) allocated by the Network Manager. These fields are there to be utilized at the User's discretion.

4.3. Technical Specifications

Slot width	1
BackPlane Current Load	0.6A
Operating Temperature	0-60 Deg C (32-140 Deg F)
Storage Temperature	-40 – 85 Deg C (-40 – 185 Deg F)
Humidity	5-95% (Non-Condensing)
Allen Bradley Platform:	ControlLogix 1756 Rack
LonWorks Connection type:	TP/FT-10
Ethernet port	10/100 MB/sec.
RS-485 port	2-wire, half duplex.
Approvals[1]	UL 60950-1:2003 CAN/CSA-C22.2 No.60950-1-03

[1] Refer to **Error! Reference source not found.** to view Certificates.

5. SlotServer Topology

The SlotServer connection varies based on the protocol being used. Refer to the appropriate driver supplement for more details. The diagram in section 4.1 shows the ports available on the SlotServer for each of the protocols.

6. Configuring the CPU interface to the SlotServer

6.1. CPU interface Description

The SlotServer Data Images (Data Arrays) share data with the ControlLogix CPU tags using the backplane for communication and the FieldServer Logix driver. To map the Logix Driver, the Driver needs to be configured in the SlotServer Configuration. The information that follows details the configuration parameters that can be used for this driver. The driver can only act as an I/O Server (Adapter) to a Control-Logix CPU.

Max Nodes Supported

SlotServer Mode	Nodes	Comments
Server	1	Only one IO image connection supported

6.2. Configuring the SlotServer as a Logix I/O Server

For a detailed discussion on SlotServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the SlotServer (See ".csv" sample files provided with the SlotServer).

The configuration file tells the SlotServer about its interfaces, and the routing of data required. In order to enable the SlotServer for Logix communications, the driver independent SlotServer buffers need to be declared in the "Data Arrays" section, the SlotServer virtual node(s) needs to be declared in the "Server Side Nodes" section, and the data to be provided to the Clients needs to be mapped in the "Server Side Map Descriptors" section. Details on how to do this can be found below.
Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

6.3. Data Arrays/Descriptors

The configuration file tells the SlotServer about its interfaces, and the routing of data required. In order to enable the SlotServer for Logix communications, the driver independent SlotServer buffers need to be declared in the "Data Arrays" section.

Section Title		
Data_Arrays		
Column Title	**Function**	**Legal Values**
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	Float, Bit, UInt16, SInt16, Packed_Bit, Byte, Packed_Byte, Swapped_Byte
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	For IO_Data_Type: REAL 1-120 INT 1-244 SINT 1-492 DINT 1-120

Example

```
//  Data Arrays

Data_Arrays
Data_Array_Name,        Data_Array_Format,    Data_Array_Length
In_1,                   Float,                76
In_2,                   Float,                76
Out_1,                  Float,                76
Out_2,                  Float,                76
```

6.4. Server Side Connection Descriptors

Section Title
Connections

Column Title	Function	Legal Values
Adapter	Adapter Name	ControlNet

Example

```
//  Server Side Connections

Connections
Adapter
ControlNet
```

6.5. Server Side Node Descriptors

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	Virtual Node ID	0-15
Protocol	Specify protocol used	Logix_SS
Timeout	IO image connection timeout	0.5s
Retry_Interval	Time to retry node online status	0.1s
Recovery_Interval	Interval to recover node to online status	0.5s

Example

```
//  Server Side Nodes

Nodes
Node_Name,          Node_ID,    Protocol,    Timeout,    Retry_Interval,    Recovery_Interval
SlotServer_CPU,     11,         Logix_SS,    0.5s,       0.1s,              0.5s
```

6.6. Server Side Map Descriptors

6.6.1. SlotServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Scan_Interval	Rate at which IO image data is updated	Use twice the rate used in RSLogix e.g. 0.2s
Function	Function of Server Map Descriptor	WRBC - to write into Input Image Data buffer RDBC - to read from Output Image Data buffer

6.6.2. Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in "Client Node Descriptor" above
IO_Data_Type	Data type of IO image	Use same as used in RSLogix to add the module INT (16-bit integer) SINT (8-bit signed) DINT (32-bit double) REAL (32-bit float)
DA_Name_Start[2]	Name of Data Array to include in IO image data	One of the Data Array names from "Data Array" section above. Data Arrays must be named as follows: DaName_x where x is a value e.g. DaName_1
DA_Count	Number of Data Arrays to include in IO image data	1-200
Protocol_Type_ID	Custom Value transferred in IO image to indicate protocol of IO image data	0 - value permitted by IO_Data_Type[3] e.g. 0 - 65535 for INT

[2] Refer to Appendix A.6 for information specific to naming Logics driver Data Arrays.

[3] The update rate decreases as the number of blocks go up. For 25 blocks we have an update rate of 25*RPI = 25*0.1 = 2.5 seconds.

6.6.3. Map Descriptors Example

Only two Map Descriptors are allowed, one to transfer data to the Logix CPU and one to accept data from the Logix CPU.

```
//    Server Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name,    Scan_Interval,  Function,  Node_Name,      IO_Data_Type,  DA_Name_Start[4],  DA_Count,  Protocol_Type_ID
Input_BP_Image,         0.2s,           WRBC,      SlotServer_CPU, REAL,          In_1,              2,         0
Output_BP_Image,        0.2s,           RDBC,      SlotServer_CPU, REAL,          Out_1,             2,         0
```

The following Data Arrays section must exist for the above Map Descriptors:

```
// Data Arrays

Data_Arrays
Data_Array_Name,    Data_Format,    Data_Array_Length
In_1,               Float,          76
In_2,               Float,          76
Out_1,              Float,          76
Out_2,              Float,          76
```

The data format should be chosen to allow no loss of resolution with the chosen IO_Data_Type of the Map Descriptor, e.g. don't use integer here if you intend transferring float data.

See the Data Arrays section for the maximum Data Array lengths allowed per IO_Data_Type chosen on the Map Descriptor.

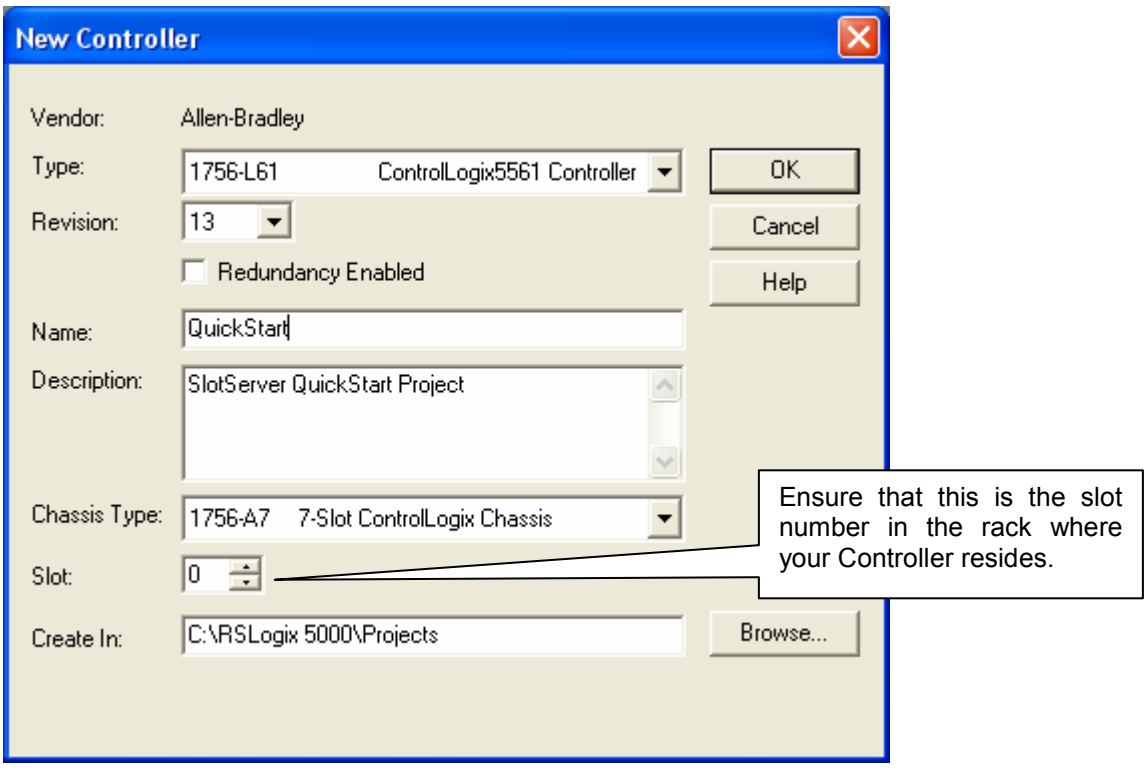
[4] Refer to Appendix A.6 for information specific to naming Logics driver Data Arrays.

7. Programming the ControlLogix CPU for a small SlotServer Interface

The discussion that follows describes the basic steps to set up and test the system for transferring data between CPU tags and the SlotServer using the I/O image method. The quick Start example uses LonWorks as the example 3rd Party Protocol. A hardcoded template is filled with Lon variables is created. Each item uses a different amount of bytes and the total adds up to 104 Lon Network Variables. This limit of 104 does not apply when using customized data items – the actual limit is 496 bytes. Refer to the Advanced Project to access more than 104 Network Variables.

7.1. Step 1: Establish an RSLogix project[5]

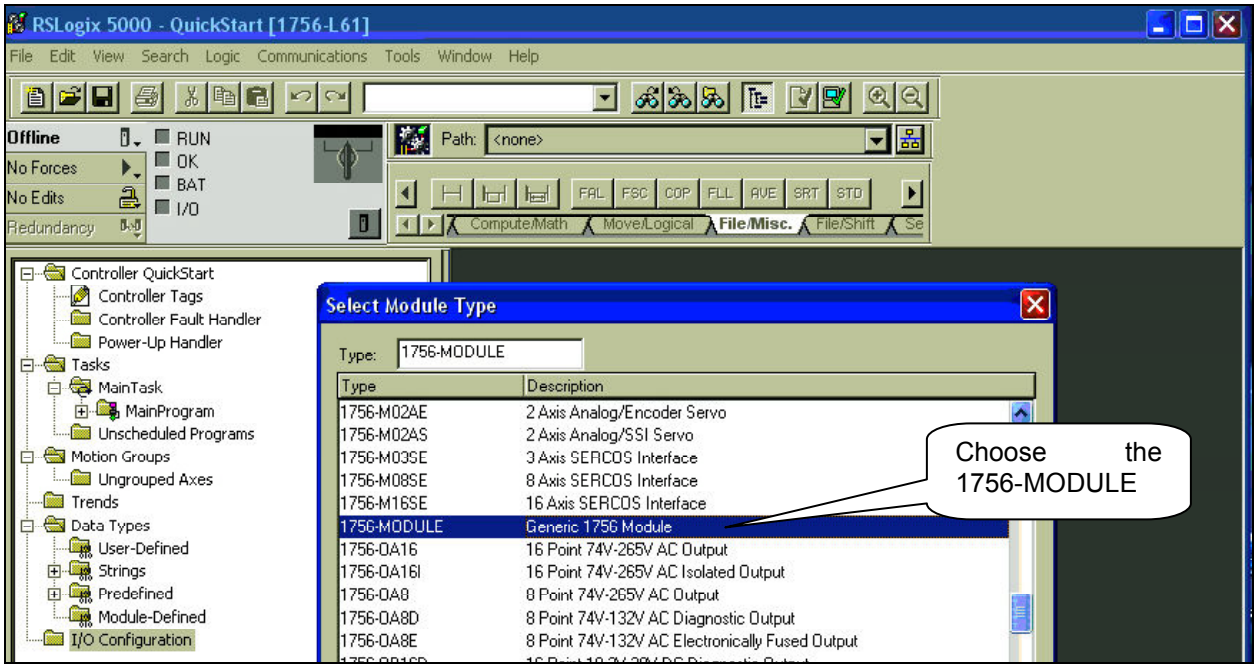
- Use File/New to create a new project or File/Open to open an existing project.

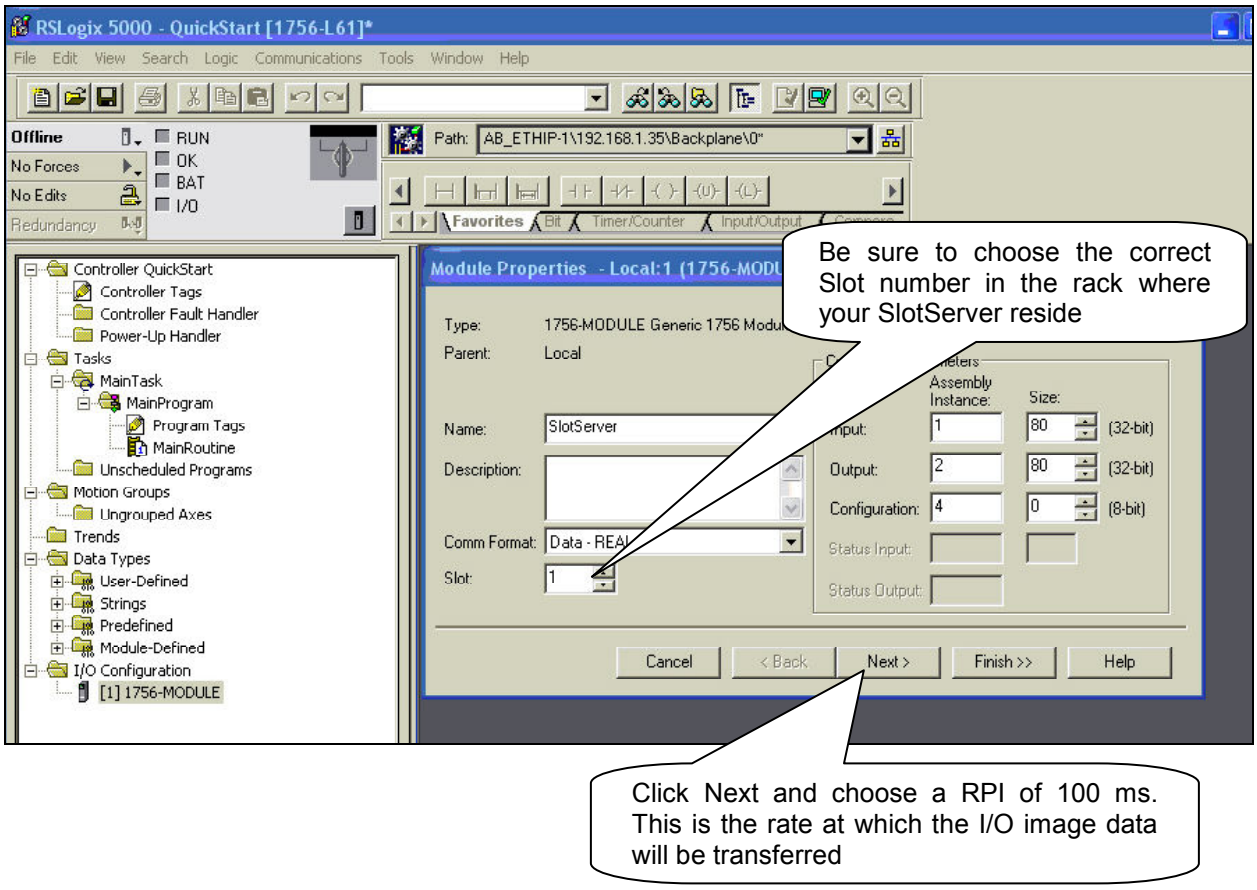


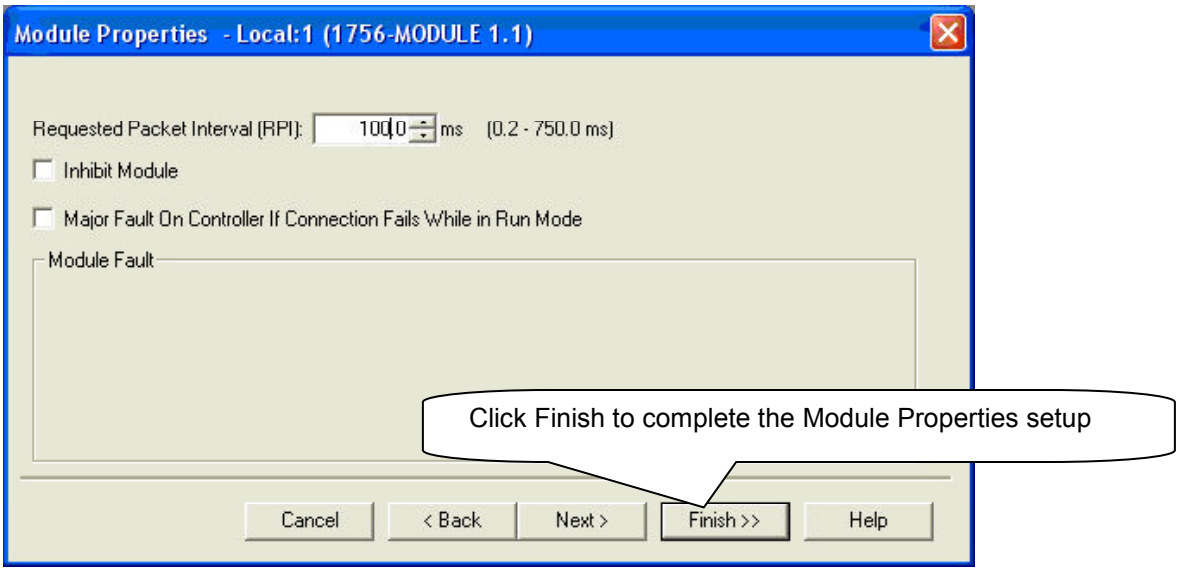
[5] Your Controller may be of a different type to the one shown in the example.

7.2. Step 2: Add and configure the SlotServer as an IO Module

- Right-click on I/O Configuration and select “New Module”.
- Choose the 1756-MODULE

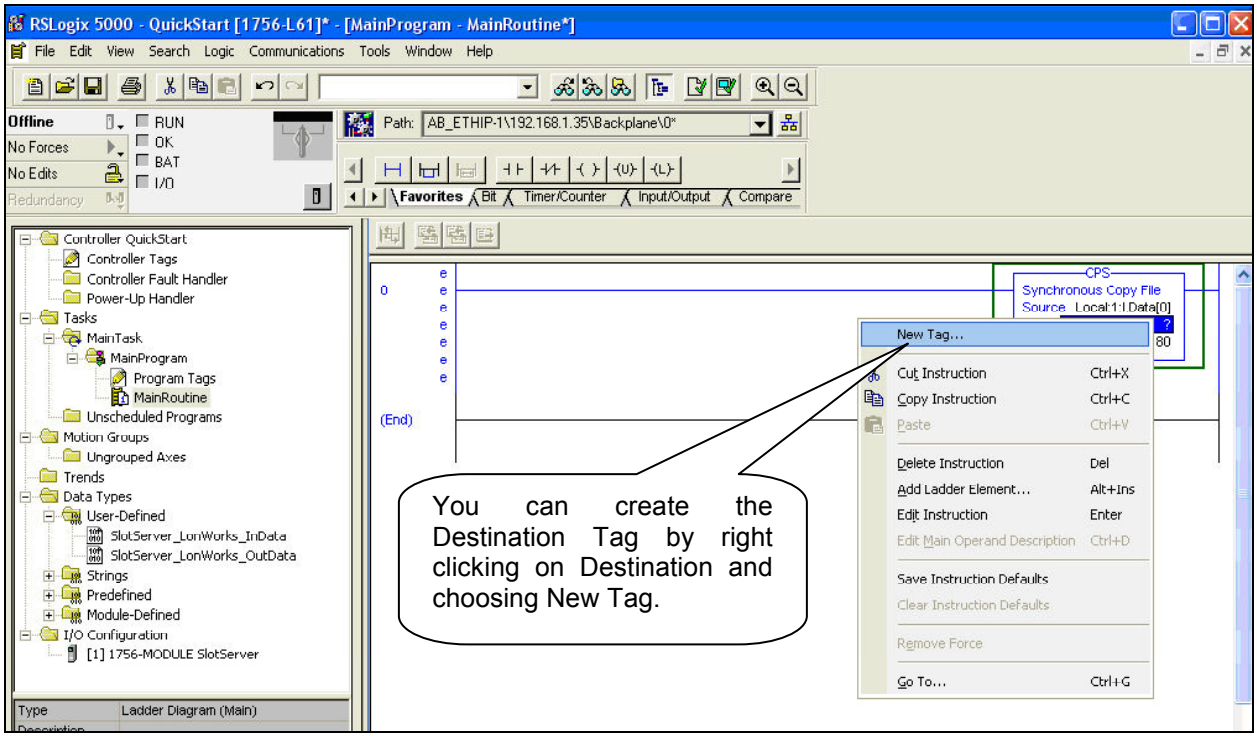


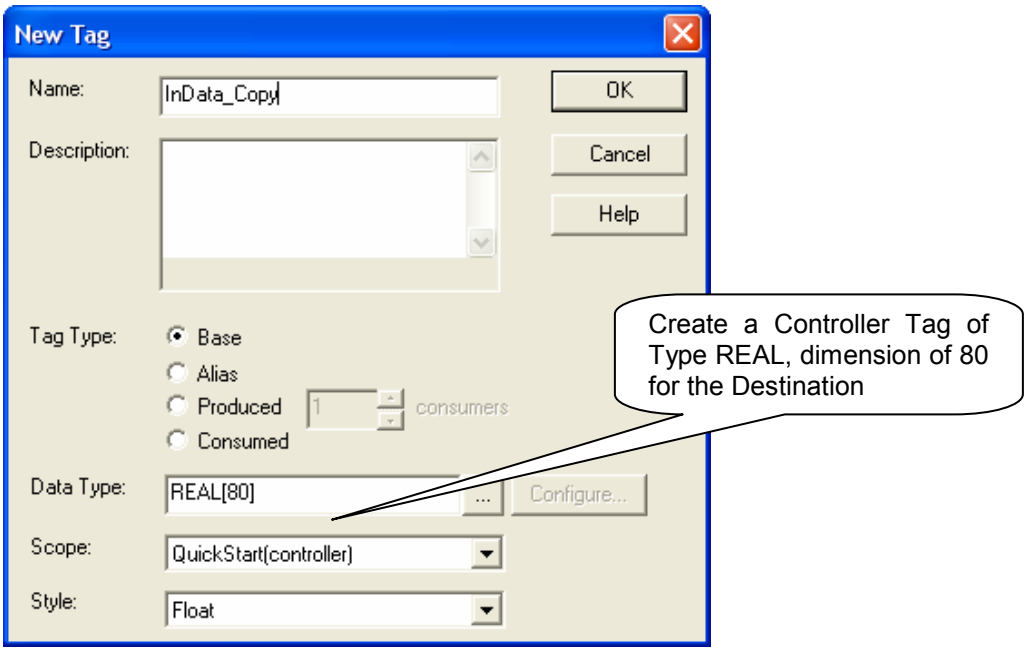




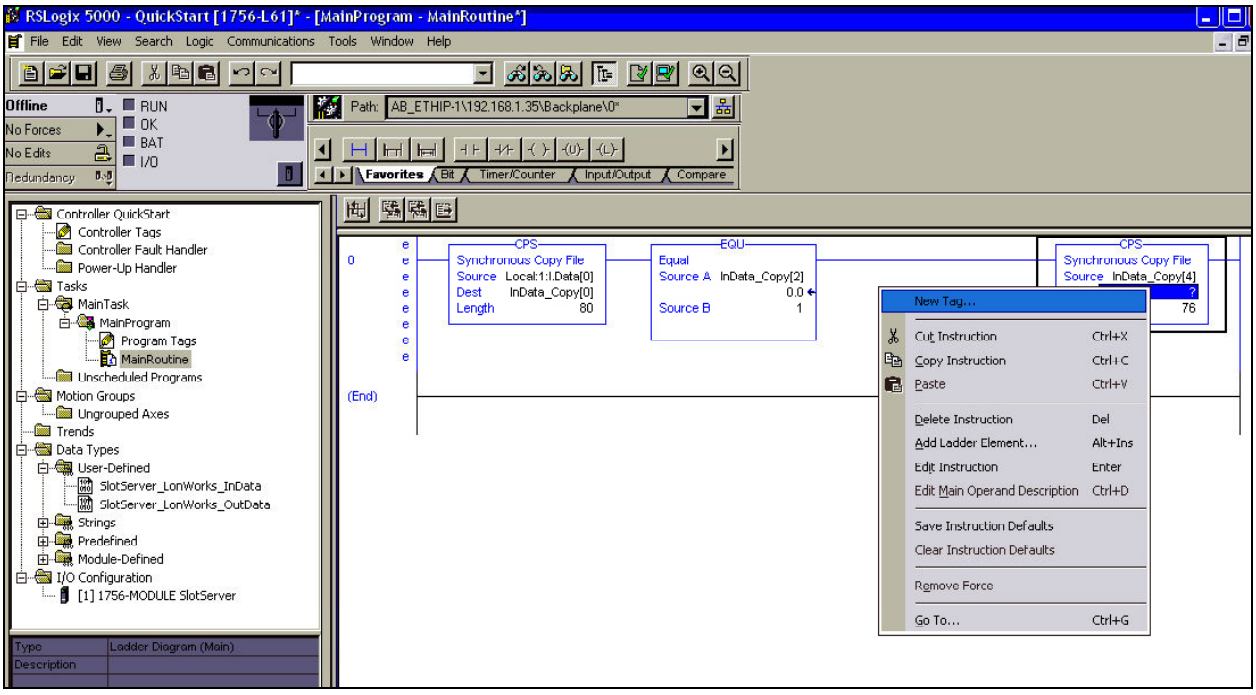
7.3. Step 3: Write Ladder Program to get Input Data from Data Arrays

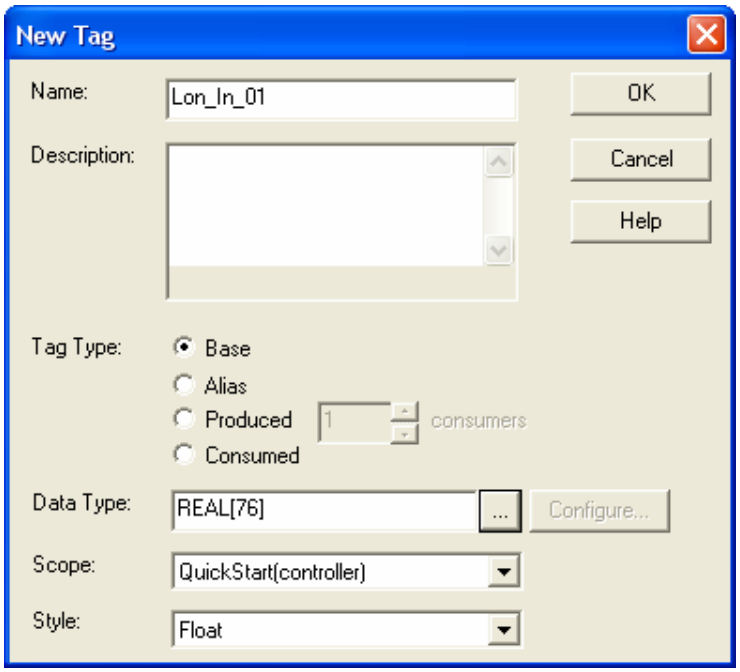
- Add a CPS (Synchronous Copy File) Ladder element to synchronize the incoming Data from the Input Data Arrays. Use the Input Image Data as Source.



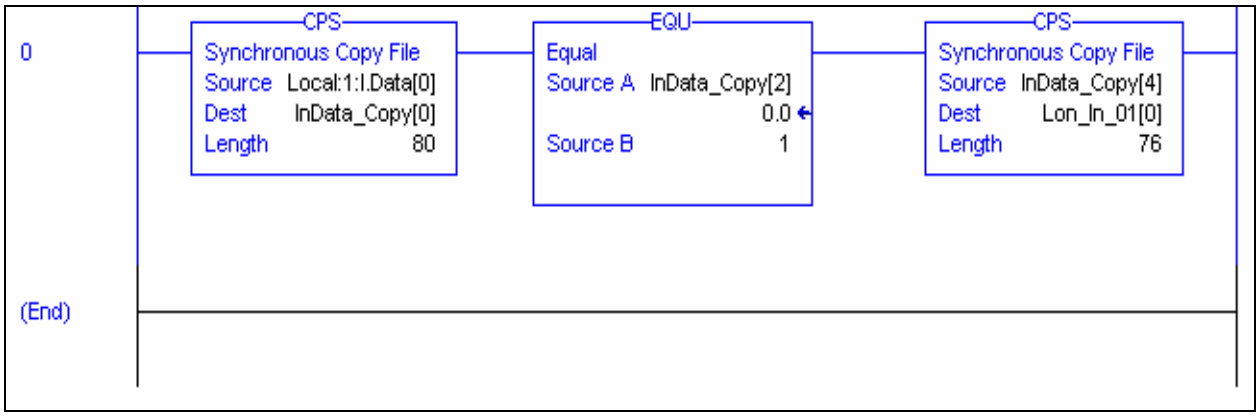


- Add an EQU (Compare if equal) ladder element to check when the first Data Array has been received. The block number is at offset 2 of the input image.
- Finally, add another CPS ladder element to copy the LonWorks Data from the InData_Copy Tag to a new Controller Tag, called Lon_In_01. Also create the Tag by right clicking on Destination and choosing New Tag. The New Tag must be of type REAL and a dimension of 76.





Below is the final ladder program to access data from LonWorks Function Block In[0]

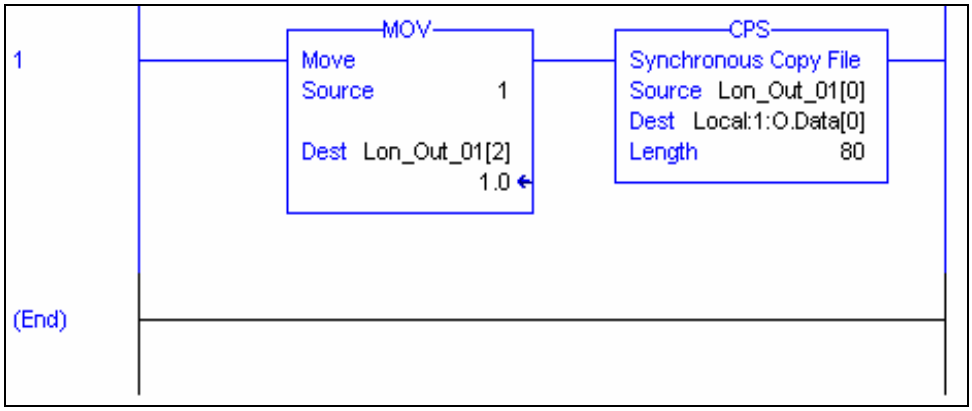


Very Important Note!
It is very important to first make a synchronous copy of the input image data before using it. If this is not done, the input data cannot be guaranteed to be from a specific SlotServer Data Array.

7.4. Step 4: Write Ladder Program to Send Output Data to Data Arrays

This step demonstrates how to write data to the Data Array Out[0]

- Create a Controller Tag called Lon_Out_01 of type REAL[80].
- Add a new rung to the Ladder program and add a MOV element to move a block number value of 1 into Lon_Out_01[2].
- Finally add a CPS (Synchronous Copy File) element to copy the full Lon_Out_01 tag into the Output Image Tag.



The LonWorks Data are present from Lon_Out_01[4] to Lon_Out_01[79]

You can create a User Defined Data Type to replace the type of Lon_Out_01 mapping the points to LonWorks point names.

Very Important Note!
It is very important to only update all the data of the Output Image Tag once using a Synchronous File Copy element. It is not permissible to update the block number into the Output Image Tag and then the data as this will cause an asynchronous transfer of data.

7.5. Step 5: Download the RSLogix Program and Run

Use the Who Active or Communications Path directly to Download and Run the Program on the Controller / CPU.

7.6. Step 6: Set up the third party connection

In this example, this step would involve binding the LonWorks variables using a LonWorks Network Manager..

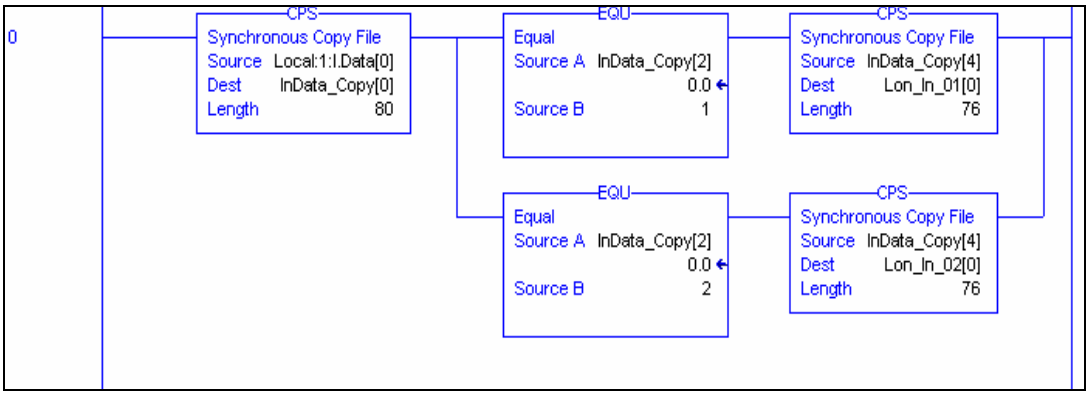
8. Programming the ControlLogix CPU for larger SlotServer Interfaces

The previous example is for accessing only one Data Array. The following steps describe how to access multiple Data Arrays.

8.1. Multiple Input Data Arrays

In this example, we access Input Data Arrays In[1] up to In[24]. We simply add to the existing ladder program as shown in the Quickstart example. Add a branch after the CPS element that copies the input image Tag and copy and paste EQU and CPS elements from the first rung. Create a new Input Tag for Lon_In_02 of type REAL and dimension 76. Finally, remember to set the EQU Source B value to 2 to compare for incoming data from the 2nd LonWorks functional block which is In[1].

See the ladder program below how to add In[1].



Very Important Note!
It is very important to first make a synchronous copy of the input image data before using it. If this is not done, the input data cannot be guaranteed to be from a specific LonWorks Function Block.

8.2. Accessing Multiple Output Data Arrays

To access more output Data Arrays is slightly more complicated since we need to create a Multiplexer in Ladder.

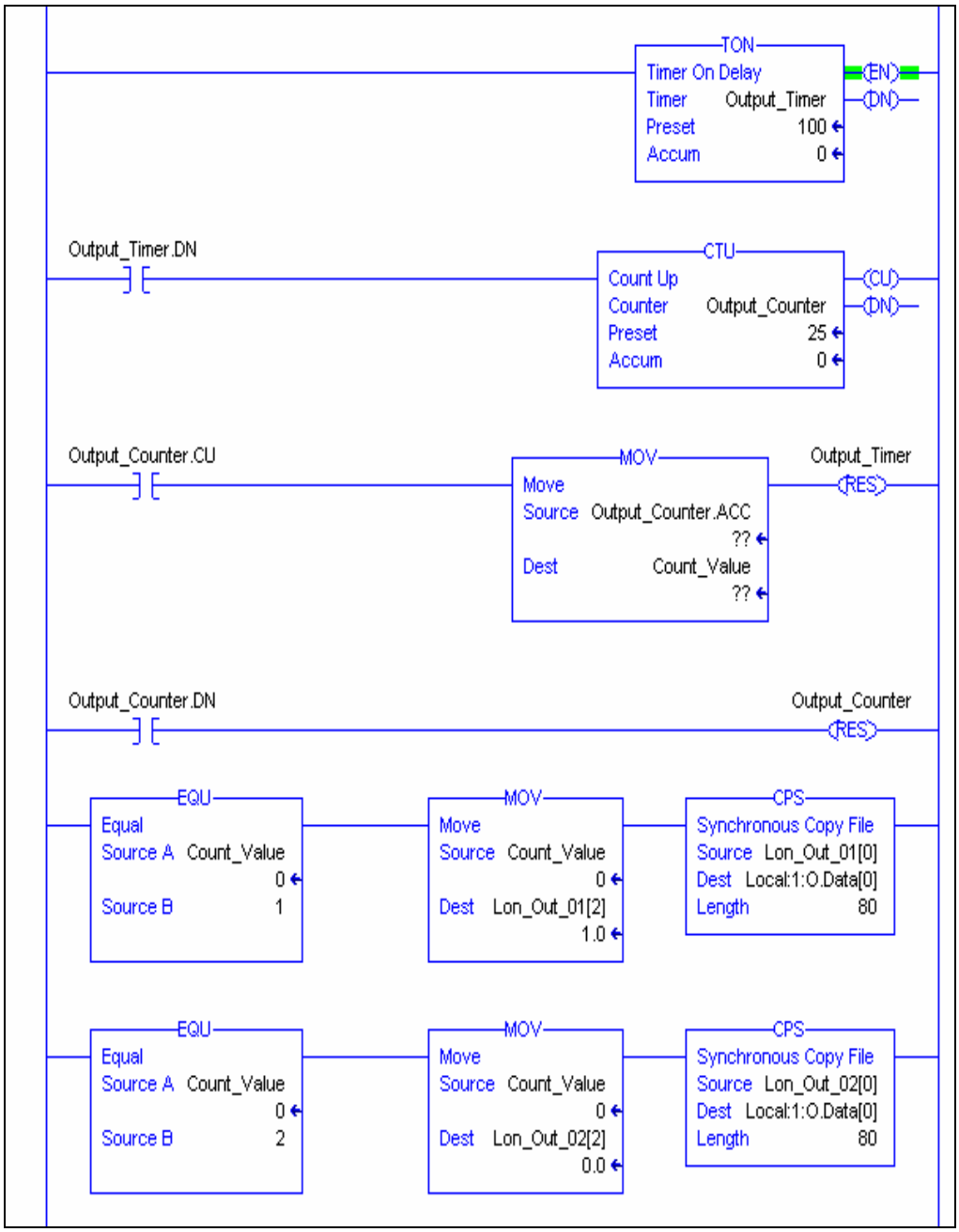
The basic steps include:

- Create a Counter which counts up every 100ms.
- Place the counter value into the Lon_Out_xx Tag at offset 2.
- Copy and the whole Tag into the output Data Image Tag for transferring to the LonWorks network.

The example program below shows an output counter that can count up to 25 which allows the transfer of data into 25 Output Function Blocks. Only 2 rungs are shown to transfer data for blocks 1 and 2. Add more rungs with more Lon_Out_xx tags to transfer data to other output Function Blocks.

It is possible to add up to 65535 blocks. The update rate decreases as the number of blocks goes up. For 25 blocks we have an update rate of 25*RPI = 25*0.1 = 2.5 seconds.

You can restrict the upcount to a certain value, e.g.2 by changing the Preset value of the CTU element.



Very Important Note!

It is very important to only update all the data of the Output Image Tag once using a Synchronous File Copy element. It is not permissible to update the block number into the Output Image Tag and then the data as this will cause an asynchronous transfer of data.

Appendix A. Advanced Topics

Appendix A.1. Description of Data Transfer Process

The data connection from the SlotServer to the Logix CPU consists of 496 bytes of input and 496 bytes of output data. Of the 2 Map Descriptors specified, the one with the WRBC function writes data to the Logix CPU filling its input data, and the one with the RDBC function accepts the Logix CPU's output data.

The Map Descriptor's IO_Data_Type field organizes the 496 bytes into either Bytes (SINT), Words (INT) or Double Words (DINT or REAL) reducing the number of elements that can be transferred. Of the resulting number of elements, the first 4 are reserved for the IO image header (Refer to Appendix A.2).

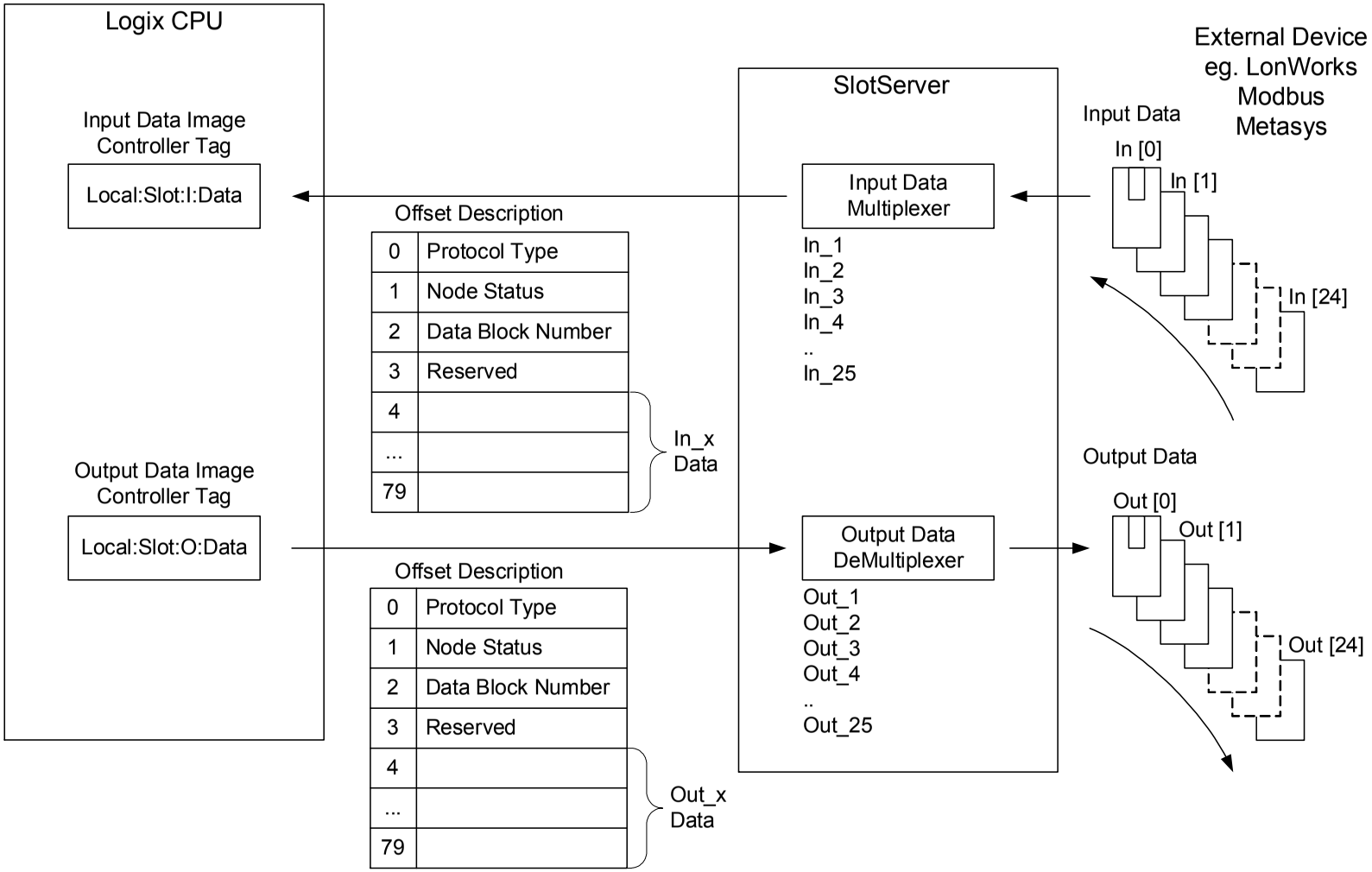
The SlotServer acts as a multiplexer when it sends data to the Logix-CPU and as a demultiplexer when it receives data from the Logix-CPU.

The diagram on the next page describes the SlotServer operation methodology:

For input data, the input data from the external device is placed into the 25 Data Arrays numbered In_1 to In_25. The SlotServer sends the data from these Data Arrays over the IO image connection one at a time by placing the block number at offset 2 of the image header and the data from offset 4. The reverse happens at the Logix-CPU where a demultiplexer is implemented in Ladder to route the data to each of the 25 CPU Tags.

For output data, the Logix-CPU has to place the data and block number into the Output Image Tag and send it to the SlotServer. The SlotServer then demultiplexes the data by placing it into the appropriate Out_x Data Array depending on the block number specified in the IO image header.

SlotServer Data Transfer over IO Data Image



Appendix A.2. The IO image header

The IO image header appears at the start of every block of image data that is transferred to or from the SlotServer to the Logix CPU. It consists of 4 items of data:

Offset into image data block	Item	Description
0	Protocol Type	The value specified under the Map Descriptor's Protocol_Type_ID field is transferred to the Logix CPU and can be used to decode the protocol. The same value has to be transferred back to the SlotServer to indicate the protocol.
1	Node Status	This field is automatically filled in by the SlotServer if a Node Status Data Array with the name SlotServerNodes is declared. Its value can be used in the Logix CPU to check the status of Nodes connected to the SlotServer.
2	Block Number	The number of the Data Array for which the IO image data is valid for, e.g. a block number of 1 will indicate the data is to or from DataArray_1
3	Reserved	Not used

Appendix A.3. How to obtain Node Status from the SlotServer

By declaring the following Data Array, the Node Status field in the IO image header will be filled in with the Node Statuses of all Nodes declared on the SlotServer:

```
//    Data Arrays

Data_Arrays
Data_Array_Name,    Data_Format,    Data_Array_Length,    Data_Array_Function
SlotServerNodes,    Bit,            256,                  Node_Status
```

Note: The Data Array Name must be as shown for this function to work correctly.

Appendix A.4. Dealing with ControlLogix RPI Settings

When setting up the SlotServer for ControlLogix, it is necessary to select the Request Packet Interval (RPI). The RPI is the rate at which data is transferred to and from the SlotServer IO buffers. The following factors need to be considered when deciding on an RPI:

1. Minimum RPI setting for the SlotServer is 100ms.
2. The Scan_Interval parameter of the two Logix Map descriptors must be larger than the RPI to ensure smooth communications and prevent timeouts.
3. The number of message blocks used does not affect the RPI setting, but will affect the effective update rate for any one message block. The effective update rate for data to/from the SlotServer's Data Arrays to the Logix CPU tags is the scan interval of the block since data is updated to the block every scan interval. Increasing the number of blocks will decrease the effective update rate per block. This update rate does not include the time taken to obtain data from the third party network, which is dependant entirely on the third party protocol involved.
4. The program scan rate should be set to run faster than the RPI. We recommend twice as fast. If the ladder program scan is slower than the RPI rate, it would be possible to miss some blocks altogether. It is further recommended that diagnostics be added to the ladder program to detect missed blocks

The effective update rate can be calculated using the following formula:

Effective update rate (EUR) = Scan interval * (number of msg block pairs[6])

e.g. (EUR) = 0.2s * 1 = 0.2s
when using only one input and output block

e.g. (EUR) = 0.2s * 25 = 5s
when using the full LonWorks Open Interface configuration.

[6] a message block pair consists of an output and input block

Appendix A.5. Installing SlotServer on a Remote Rack using CNB Cards

Appendix A.5.1. Hardware and Software requirements

In order to perform this application, the following hardware and software is required as a minimum:

- At least two 1756 racks, where one rack contains the CPU and the other rack contains the SlotServer
- Two Controlnet CNB cards (with cable for connection) to connect the racks to each other
- RSlinx
- RSLogix
- RSNetworx
- SlotServer EDS file (Available on SlotServer CD, or call Technical Support)

Note that it is possible to connect to SlotServer on a remote rack using other 1756 bridging cards. This chapter only deals with CNB, but the principles for using other bridging cards are similar.

Appendix A.5.2. Setup

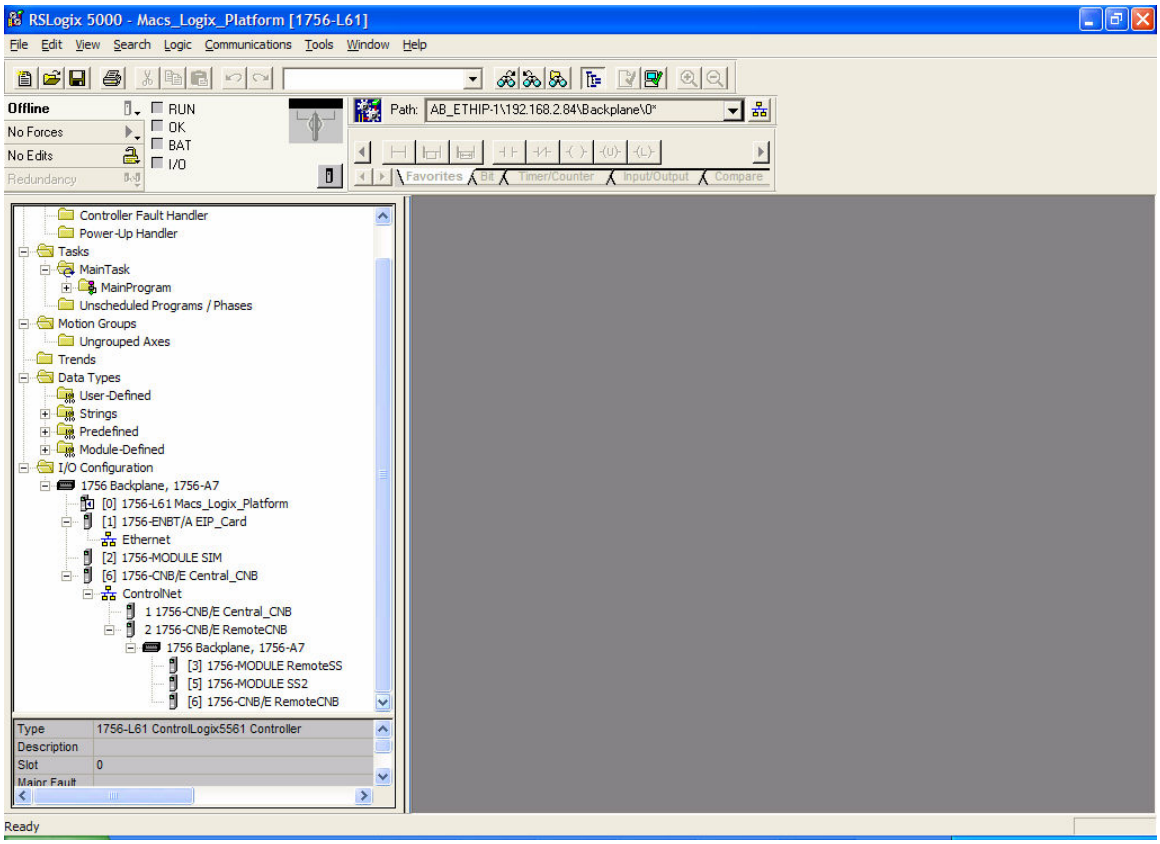
- Install the CPU and the first CNB card in the local rack.
- Install the second CNB card and the SlotServer in the remote rack.
- Connect the ControlNet Network
- Power up the racks
- Install the SlotServer EDS File using the RSLinx Hardware EDS Installation Tool.
- Make sure the SlotServer has a valid configuration loaded. The default configuration shipped with the SlotServer should suffice.

Appendix A.5.3. RSLogix configuration

In order to see the SlotServer from the CPU, the Hardware must be configured in RSLogix as follows:

- Configure the cards in the local rack in the I/O Configuration section, including the CPU and the local CNB card.
- Right click on the local CNB card and add the remote CNB card using the “New Module” function.
- Right Click on the remote CNB card, and add the 1756 Backplane
- Right Click on the 1756 Backplane and add the cards that are present in the remote rack, including the SlotServer (As a generic I/O module-see earlier section on how to do this)
- Save the RSLogix configuration, and download it to the PLC.

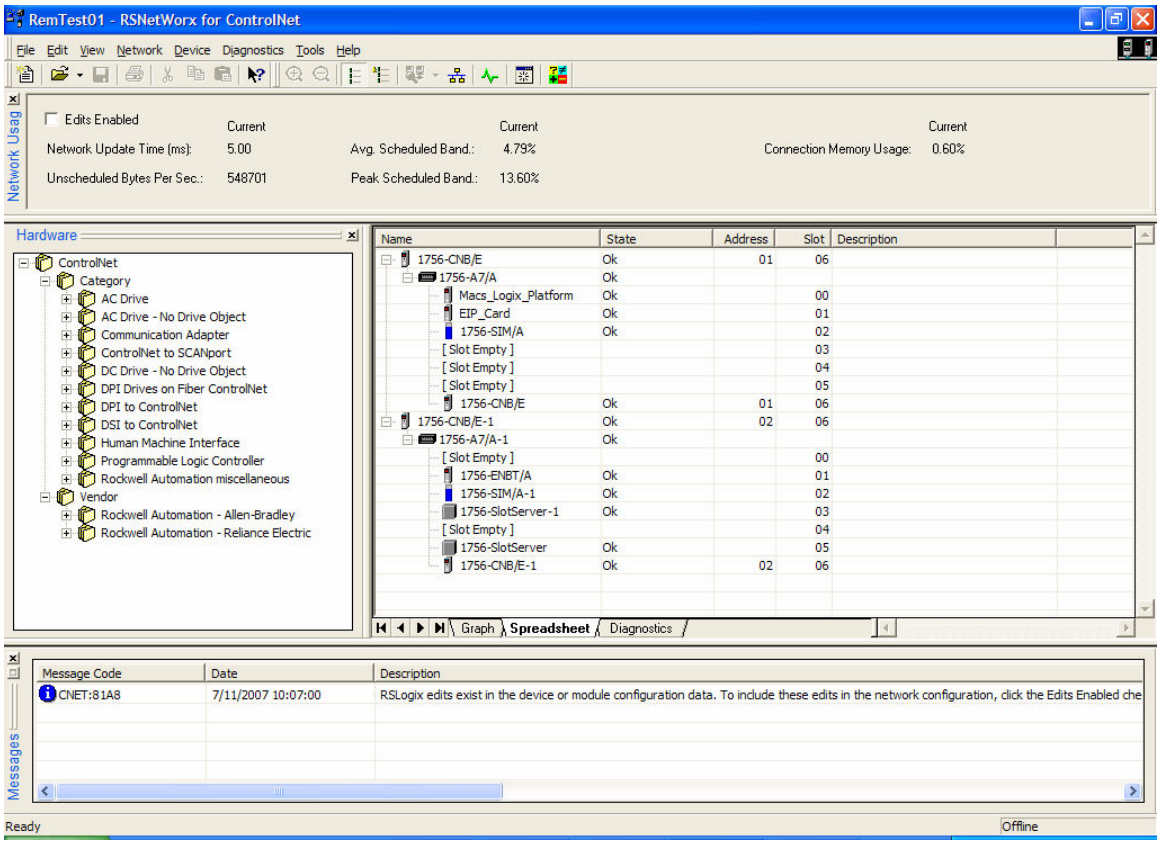
The finished I/O configuration should look similar to this:



Appendix A.5.4. RSNetWorx configuration

- Open up RSNetWorx and add the two CNB Cards to the Network by dragging them onto the Network in the Graph tab (Must be done with Edits Enabled). Follow the prompts on the screen to configure the Chassis being used, and the cards in each Chassis (Rack).
- Go Online with RSNetWorx, and then press “save”. This will transfer the RSNetWorx Configuration to the Keeper.

The final RSNetworx Configuration should look similar to this:



Appendix A.5.5. Testing

The SlotServer should now be visible to the CPU. Go Online with RSLogix and check the Input buffer of the SlotServer for data. A good check is to examine offset 2 of the input tag for a non-Zero value. If the SlotServer is multiplexing (DA_Count >1), then this value will be cycling through the Buffer numbers, otherwise if DA_Count=1, then offset 2 will be fixed at 1. If offset 2 is zero, then the SlotServer is not being seen by the CPU, and Diagnostics will need to be performed using RSNetWorx and RSLogix to determine the cause of the problem.

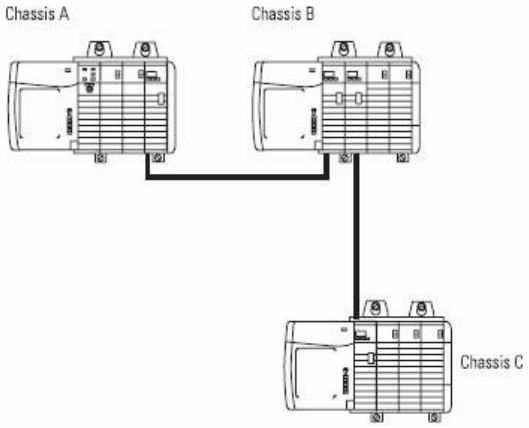
Appendix A.5.6. Connection limitations -Controlling the SlotServer using ControlNet

- Only one remote I/O rack is supported.
- I/O can only be added online using a direct connection.

The following Vendor information provides clarification:

For a ControlLogix controller to control 1756 I/O, the I/O must be:

- in the same chassis as the controller **or**
- on a ControlNet network that is local to that controller **or**
- on an Ethernet/IP network that is local to that controller



For example, assume that the network links in this example are either ControlNet or EtherNet/IP links. Both links can be the same, or one link can be a ControlNet link and the other can be an EtherNet/IP link. Chassis A can control the 1756 I/O modules in Chassis A and in Chassis B, but not in Chassis C. The ControlLogix controller in Chassis A can only send messages to the devices in Chassis C.

Add I/O online

When online:

- You can add 1756 I/O modules to the local chassis, remotely via the unscheduled portion of a ControlNet network, and remotely via an EtherNet/IP network.
- The I/O modules you add online use direct connections (rack-optimized connections are not supported when adding I/O modules online).

Appendix A.6. Rules for Naming Logix driver Data Arrays

Unlike most other FieldServer drivers, the Logix driver attaches significance to the name of the Data Array used in the Logix Driver Map Descriptor. This is done to allow the user to easily declare a series of Data Arrays to be multiplexed through the input and output buffers of the SlotServer. Most users will probably stick to a naming convention where the Data Arrays are named In_1 to In_x for input buffer arrays, and Out_1 to Out_y for output buffer arrays (where x and y are numbers reflecting the maximum input and output Array numbers respectively).

For example, an application that multiplexes 6 Data Arrays worth of data through the Input buffer will probably use data arrays named In_1 though In_6. In this example, DA_Name_Start is declared as In_1, and DA_Count is declared as 6.

It is not absolutely necessary to use the naming convention described above, however, and while the user has some latitude for declaring names, the following restrictions must be understood before attempting a different naming convention:

- The Data Array name must end in _x, where x is a positive integer number.
- The total length of the Data Array name (including _x) must not exceed 15 characters.
- No leading zeros should be used in the _x number (For example, use _5, not _05)
- The “x” part of the _x in the data array name will be the number shown in offset 2 of the input buffer for the purposes of de-multiplexing in the CPU.
- There can only be one Map Descriptor for linking Data Arrays to the input buffer, and one Map Descriptor for linking Data Arrays to the output buffer (Using wrbc in the function parameter links the Map description to the input buffer, and using rdbc links the map description to the output buffer). This means that trying to map Data Array number sequences that are not continuous will not be possible (For example, you can map numbers 5 through 25, but you cannot map numbers 1 through 3, and then 5 through 8 at the same time).
- The Data Arrays must be declared individually in the Data Arrays Section. Declare all data arrays, not just the start Data Array.

The following examples describe legal and illegal naming conventions respectively:

Example 1: Legal Map Descriptors:

Map_Descriptors						
Map_Descriptor_Name,	Scan_Interval,	Function ,	Node_Name,	IO_Data_Type,	DA_Name_Start,	DA_Count
Input_BP_Image,	0s,	WRBC,	CPU1,	INT,	Test_5,	2
Output_BP_Image,	0s,	RDBC,	CPU1,	INT,	Test_3,	2

Example 2: Illegal Map Descriptors:

Map_Descriptors						
Map_Descriptor_Name,	Scan_Interval,	Function,	Node_Name,	IO_Data_Type,	DA_Name_Start,	DA_Count
Input_BP_Image,	0s,	WRBC,	CPU1,	INT,	Test_05,	2
Output_BP_Image,	0s,	RDBC,	CPU1,	INT,	Test,	2
Output_BP_Image2,	0s,	RDBC,	CPU1,	INT,	Test_-3,	-5
Output_BP_Image3,	0s,	RDBC,	CPU1,	INT,	Test6,	2

Appendix B. Troubleshooting tips

Appendix B.1. Things to check when communications fail.

- Check for loose cabling on the third party network
- Verify that the correct program is loaded to the CPU
- Verify that the correct data types for the tags have been used.

THIS PAGE INTENTIONALLY LEFT BLANK